

AD-A156 059

A QUICKLY TESTED PASCAL RANDOM NUMBER GENERATOR FOR
MICROCOMPUTERS(U) MITRE CORP BEDFORD MA
C J COLWELL ET AL. MAY 85 MTR-9067 ESD-TR-84-202

1/1

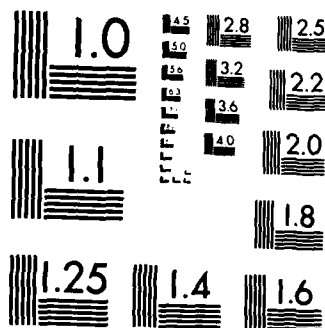
UNCLASSIFIED

F19628-84-C-0001

F/G 9/2

NL

									END				
									FILED				
									DTIC				



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

2

ESD-TR-84-202

MTR-9067

AD-A156 059

A QUICKLY TESTED PASCAL RANDOM NUMBER GENERATOR
FOR MICROCOMPUTERS

By

C. J. COLWELL
R. A. DRAMSTAD
M. E. LOPEZ

MAY 1985

Prepared for
DEPUTY COMMANDER FOR AIRBORNE WARNING AND CONTROL SYSTEMS
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
Hanscom Air Force Base, Massachusetts



DTIC FILE COPY

DTIC
ELECTE
JUL 05 1985
S D
H G

Approved for public release
distribution unlimited

Project No. 4110
Prepared by
THE MITRE CORPORATION
Bedford, Massachusetts
Contract No. F19628-84-C-0001

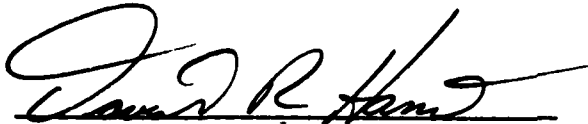
85 6 11 001

When U.S. Government drawings, specifications or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Do not return this copy. Retain or destroy.

REVIEW AND APPROVAL

This technical report has been reviewed and is approved for publication.



DAVID R. HARRIS, 2Lt
Software Engineer



LOUIS D. MASIELLO
Deputy Director of Engineering
Deputy Commander for AWACS

FOR THE COMMANDER



CHARLES W. ALLPORT, Colonel, USAF
Assistant Deputy Commander for Airborne
Warning and Control Systems

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

AD-A156059

REPORT DOCUMENTATION PAGE				
1a. REPORT SECURITY CLASSIFICATION Unclassified		1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.		
2b. DECLASSIFICATION DOWNGRADING SCHEDULE				
4. PERFORMING ORGANIZATION REPORT NUMBER(S) MTR-9067 ESD-TR-84-202		5. MONITORING ORGANIZATION REPORT NUMBER(S)		
6a. NAME OF PERFORMING ORGANIZATION The MITRE Corporation	6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION		
6c. ADDRESS (City, State and ZIP Code) Burlington Road Bedford, MA 01730		7b. ADDRESS (City, State and ZIP Code)		
8a. NAME OF FUNDING/SPONSORING ORGANIZATION (see other side)	8b. OFFICE SYMBOL (If applicable) YW	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F19628-84-C-0001		
8c. ADDRESS (City, State and ZIP Code) Electronic Systems Division, AFSC Hanscom Air Force Base, MA 01731-5000		10. SOURCE OF FUNDING NOS.		
11. TITLE (Include Security Classification) A QUICKLY TESTED PASCAL RANDOM (cont.)		PROGRAM ELEMENT NO.	PROJECT NO. 4110	TASK NO.
12. PERSONAL AUTHOR(S) Colwell, C. J., Dramstad, R. A., Lopez, M. E.		WORK UNIT NO.		
13a. TYPE OF REPORT Final	13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day) 1985 May		15. PAGE COUNT 53
16. SUPPLEMENTARY NOTATION				
17. COSATI CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB. GR.		
			Microcomputers	
			Pascal	
			Random numbers	
			Random variate	
19. ABSTRACT (Continue on reverse if necessary and identify by block number) An assembly language subroutine that uses a shifting mask to generate pseudo-random numbers has been written for use with Apple II microcomputer Pascal. The routine is nearly 80 times faster than a direct Pascal multiplicative congruence scheme. With the recommended starting parameters, the resulting stream has passed 10 different statistical tests for density and randomness. An efficient extension to the normal distribution is also presented.				
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT <input checked="" type="checkbox"/> DTIC USERS <input type="checkbox"/>		21. ABSTRACT SECURITY CLASSIFICATION Unclassified		
22a. NAME OF RESPONSIBLE INDIVIDUAL Diana F. Arimento		22b. TELEPHONE NUMBER (Include Area Code) (617)271-7454	22c. OFFICE SYMBOL Mail Stop D230	

DD FORM 1473, 83 APR

EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

- 8a. Deputy Commander for Airborne Warning and Control Systems
- 11. NUMBER GENERATOR FOR MICROCOMPUTERS

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

ACKNOWLEDGEMENTS

This document has been prepared by The MITRE Corporation under Project No. 4110, Contract No. F19628-84-C-0001. The contract is sponsored by the Electronic Systems Division, Air Force Systems Command, Hanscom Air Force Base, Massachusetts.

Accession For	
NTIS GRAM	7
DTIC TAB	51
Unannounced	
JUL 1968	
Re:	
Distribution	
AS	15
AL	20
ALL INFORMATION CONTAINED HEREIN IS UNCLASSIFIED	
DATE 11-19-81 BY SP-6 BTJ/KJS	



TABLE OF CONTENTS

<u>Section</u>	<u>Page</u>
1 INTRODUCTION	1
2 IMPLEMENTATION NOTES	3
ENHANCEMENTS	5
EXTENSION TO THE NORMAL DISTRIBUTION	6
3 STATISTICAL TESTS	7
GENERAL	7
MEAN AND STANDARD DEVIATION	8
FREQUENCY AND KOLMOGOROV-SMIRNOV TEST	9
DISTRIBUTION OF RANDOM PAIRS	10
PERMUTATION OF RANDOM TRIPLETS	11
GAP TEST	12
"POKER" TEST	14
SERIAL CORRELATION TEST	15
RUNS TEST	17
4 DISCUSSION OF RESULTS	19
LIST OF REFERENCES	21
APPENDIX A PROGRAM LISTING	23
APPENDIX B LISTINGS OF TEST PROGRAMS	25
APPENDIX C FREQUENCY DISTRIBUTION AND KOLMOGOROV-SMIRNOV TEST	45
APPENDIX D OBSERVED DISTRIBUTION OF RANDOM PAIRS	47

SECTION 1

INTRODUCTION

"An applied mathematician is a person that produces usable algorithms."

J. B. Rosser, 1967

The generation of a deterministic series of variates with a prescribed frequency distribution is a task of capital importance in simulation work. Having access to a subroutine that generates a deterministic stream of random uniformly distributed numbers is an essential requisite in this task, since any computable distribution can be generated from such a stream of random numbers (Kahn, 1956).

The property of randomness, or random behavior, of any part of a deterministic stream of random numbers r_1, r_2, \dots, r_n is also a key requirement. Due to the conflicting requirements of true randomness and deterministic generation, computer generated random numbers are generally referred to as pseudo-random numbers.

There are many established techniques for creating a deterministic stream of random numbers (Hull and Dobell, 1962). The purpose of this paper is to present a technique that is fast, is suitable for small microcomputers, and has passed a series of stringent statistical tests for randomness.

Section 2 discusses the implementation of the program developed for the Apple II computer which uses a 6502 microprocessor and listed in appendix A. Section 3 presents the results of statistical tests on the randomness of the stream generated when the recommended starting parameters are used, and section 4 discusses the results.

SECTION 2

IMPLEMENTATION NOTES

Smith's third principle: "Never do anything for the first time."

The proposed random number generator is based on the work of Tausworthe and on an algorithm suggested by Foster (1978); its implementation depends on the data structure used for reals by the microcomputer used. The Foster algorithm provides a technique for generating a fast random bit stream; the internal data structure of real numbers in Pascal dictates where these bits should be stored. The Tausworthe algorithm has been implemented as suggested by Foster (ibid). A 32-bit long register is created initially using a pattern of ones and zeroes from a combination of prime numbers. When a new random number is needed and the leftmost bit of the pattern is a zero, the contents of the register are shifted left one place and the carry bit is rotated into the least significant bit (LSB) of the register. If the leftmost bit is a one, an exclusive-or (XOR) operation is performed first on the register using a 4-byte mask suggested by Foster; then the contents of the register are shifted using the same technique.

In the present implementation, the exponent and sign bits were initially set by making the real random number RNB in the calling program equal to one. Usually, the exponent requires one byte and the sign one bit; therefore, only twenty-three bits are shifted using the process described above, leaving the eight exponent bits unchanged. This creates a random number between one and two in the shifting register that is later normalized in the Pascal program by subtracting unity from it.

Real numbers in most microcomputers are stored using 32 bits or four bytes. From the most significant to the least significant bit, a real number is constructed of a single sign bit, an 8-bit excess 127 exponent field, and a 23-bit mantissa. There is also an implied bit, set to 1, between the exponent and the mantissa; this means that all bit patterns in a real number's mantissa are legal and normalized. An explanation of this widely used real number format can be found in Kellner (1980).

Since unit random numbers lie in the range from zero to unity, this range includes many integer powers of 2, i.e., 0.5 or 2^{-1} , 0.25, or 2^{-2} , etc., so the exponents of all representable random numbers can range over a fairly large interval. To avoid the complexities of normalizing the mantissas and changing exponents, the random number was initialized to 1.0. This clears the sign bit and sets the exponent so that the implied bit (with a zeroed mantissa) represents $+2^0$. Therefore, when the lower 23 bits of the real number (the variable RNB in the listing, section 2) are manipulated, any resulting combination yields a real number in the range 1.0 to 2.0.

These manipulations must be conducted in assembly language unless the computer can perform 32-bit logical operations. The one part that can be done very efficiently in Pascal is subtracting 1.0 from the assembly language-generated random number which brings RNB into the desired range of zero to unity. This subtraction accomplishes both the necessary normalization of the mantissa and the shifting of the exponents of the received assembly language real number.

The way Pascal and assembly language are linked in the microcomputer affects the subroutine code. The proposed assembly language routine RNUM deals with three four-byte data structures: first, the 32-bit masking register, MASK, which remains unchanged from one call to RND to the next; second, the 32-bit feedback shift register, SEED, which changes with each call and which must be preserved between calls; and finally, the variable RNB that has to be accessed by both Pascal and the assembly language routines.

In the Apple microcomputer, variables stored in assembly language routines that use the .BYTE or .WORD directives are lost between calls to the routine, a behavior analogous to that of local variables in Pascal which are undefined at the start of the procedure's execution. In order to initialize MASK, SEED, and RNB; to give the RNUM assembly language procedure access to these variables; and to preserve their values between various invocations, these variables must be defined as global in the calling Pascal program. They are accessed by the assembly language subroutine by using the host-communication directive .PUBLIC (1980 Apple Manual, pp. 166-167). These external references must be resolved by the Linker (1980 Apple Manual, pp. 186-193).

An important point is the initialization of the subroutine. Various statistical tests, to be described in section 4, were used to select the initial prime numbers for the SEED that would provide a stream with the desired properties. The values

RNB = 1.0	MASK[0] = 83
SEED[0] = 97	MASK[1] = 181
SEED[1] = 101	MASK[2] = 118
SEED[2] = 103	MASK[3] = 0
SEED[3] = 107	

provided the best results and are recommended. If any of these numbers are changed, prime numbers should be substituted. The efficient use of subroutine calls and an extension to the normal distribution are covered in the following enhancement notes.

ENHANCEMENTS

Two obvious enhancements are possible. The first one is to convert the proposed routine into a library Unit (in Apple Pascal); the other is a randomization procedure to ensure a different start for the stream each time the procedure is called.

It seems convenient to implement RND as a function inside a library Unit residing in SYSTEM.LIBRARY (1980 Apple Pascal Manual, pp. 75-81). In this way, the function RND could be used after including the Unit name in a "USES" line at the start of the program (1980 Apple Pascal Manual, p. 72). This yields two distinct advantages: implementation details are hidden and the external routine is prelinked. The recommended initializations could also be performed by the initialization block of the Unit. Moreover, the variables RNB, SEED, and MASK would be hidden in the implementation part of the Unit, as well as the external subroutine RNUM. None of these routines would be accessible from the host program, and only the function RND would remain visible due to its inclusion in the interface part of the Unit.

As written in section 2, the function RND provides a repeatable deterministic stream of pseudo-random numbers; that is, each time the program is run, the same sequence of numbers is generated. This is not necessarily a disadvantage, especially during the debugging stage when duplicating previous results is desirable. Also, this repeatability ensures that the generated stream has the desired statistical properties (section 4). To obtain a random different starting point, the following code could be used:

```
Procedure Rndize; (*equivalent to the APPLESTUFF procedure RANDOMIZE*)
```

```
VAR ch: char;
```

```
BEGIN
```

```
  Writeln("Press any key to continue");
```

```
  While not KEYPRESS do RNUM;
```

```
  READ(ch)
```

```
END;
```

The only disadvantage of this routine is that it requires using the "Applestuff" library Unit in order to use the function "KEYPRESS." This is not a problem unless the function RND is also in a library Unit, when nested "Uses" clauses would become necessary.

EXTENSION TO THE NORMAL DISTRIBUTION

The other very frequently required distribution is the random unit normal. The following Pascal subroutine generates a unit normal random deviate, RNORMAL. It assumes that the function RND is available, as described in section 2.

REPEAT

```
U: = RND;  
V: = RND;  
X: = 1.715528*(V-0.5)/U;  
RNORMAL: = X;  
X2: = X * X;  
IF(X2<=5 - 5.136102*U) THEN EXIT RNORMAL;
```

UNTIL(X2<=1.4+1.036961/U) AND (X2<=-4/LN(U));

This algorithm is recommended since it minimizes the need for computing transcendentals and produces deviates that are more normally distributed (especially at the tails) than those from the widely used technique of averaging 12 random unit deviates. The average computing time for one random normal deviate is approximately equal to that required for executing 2.74 RND functions, 2.74 multiplications, 1.84 additions, .70 divisions, and 0.23 logarithms.

SECTION 3

STATISTICAL TESTS

"Nec Babylonios temptaris numero"
(Never trust random numbers)

GENERAL

Most random number generators cannot be adequately tested, even in theory (Knuth, 1981). In practice, a stream of pseudo-random numbers is generated, and selected "statistical tests" are conducted on the stream. All quantitative attempts to qualify "random behavior" are difficult. In fact, if too many tests are used and too strict a definition of "randomness" is insisted upon, the surprising conclusion is that there is no such thing as a truly random sequence! (Knuth, *ibid.*) Wiener has emphatically stated that "the advantage of long runs of statistics under widely varying conditions is specious and spurious." For if one looks long and assiduously enough for the unusual, one is bound to find it! Two goals are particularly important when testing random number generators: first, to screen "bad" sequences early, and second, to apply only those tests that are useful for the purposes that the numbers will be used. In general, the objectives are to obtain a stream that is both "dense" and "efficient." A great deal of the work in this area is more art than science.

Knuth (1981) has discussed in detail the art of empirically testing random numbers and has pointed out some examples of horrendously poor random streams that are still widely used. In fact, the present paper was motivated by the bad behavior of a random number generator; this bad behavior was not discovered until some "unusual" results were obtained in actual simulation work. Crigler and Shields (1982) (Naval Surface Weapons Center) have adapted Knuth suggestions into a comprehensive FORTRAN program that performs eleven different statistical tests. Some of the tests used by Crigler and Shields (*ibid*) are very involved and, in view of Wiener's dictum, probably redundant for most purposes.

We have chosen ten of the tests recommended by Knuth (*ibid*) that are efficient and relatively easy to program. In order to interpret the results of these tests properly, it is important to recognize that they are applied to a finite-sized, albeit large, sample of outputs from our "random number generator." Thus, even should the elements in the sample

APPENDIX A

PROGRAM LISTING

"If the applied mathematician is going to succeed, he must from time to time turn temporarily into whatever sort of scientist he is getting an answer for, and try out the proposed algorithm to see if it is really usable."

J. B. Rosser, 1967

Following is a listing of an assembly language subroutine developed for the Apple II microcomputer. The program has been run successfully and is approximately 80 times faster than a previous congruence program written directly in Pascal. A description of the algorithm and other pertinent implementation notes are presented in section 3.

.PROC RNUM,0

```
;
;RNUM is an Assembly Language subroutine to generate random numbers and
;pass them back to a Pascal calling program.
;
;The following algorithm was used*:
;First, seed a 32-bit-long shift register with a pattern of ones and
;zeros representing a prime number. If the leftmost bit is a zero then
;shift the contents of the register left one place, and rotate the carry
;bit into the least significant bit (LSB) of the random number. If the
;leftmost bit of the seed is a one take the exclusive-or of the contents
;of the shift register with a 4-byte mask suggested by Foster. (See
;Pascal calling routine.) Shift the contents of the register left one
;place and rotate the carry bit into the LSB of the random number.
;
;The exponent and sign bits were set by placing a 1.0 in the real number
;RNB in the calling program. Twenty-three bits were shifted into the 32
;bit random number leaving the sign bit and 8 exponent bits unchanged.
;
;The random number RNB, the seed, and mask are all defined in the Pascal
;calling program and are shared with the assembly subroutine by use of
;the host-communication directive .PUBLIC. These external references
;must be resolved by the linker.
;
;*Foster, Caxton C., PROGRAMMING A MICROCOMPUTER : 6502, Reading,
;Massachusetts, Addison-Wesley Publishing Company, 1978.
```

LIST OF REFERENCES

1. Apple Pascal, 1980, "Language Reference Manual," Cupertino, CA: Apple Computer Inc., pp. 209.
2. Bennet, C. A. and N. L. Franklin, 1954, "Statistical Analysis in Chemistry and the Chemical Industry," NY: John Wiley and Sons.
3. Crigler, J. R. and P. A. Shields, 1982, "Random: A Computer Program for Evaluating Pseudo-Random Number Generators," Tech Rep NSWC TR-82-93, Alexandria, VA: Defense Tech. Info. Center, pp. 40, (AD A118 412).
4. Foster, Caxton C., 1978, "Programming a Microcomputer," Reading, MA: Addison-Wesley.
5. Hull, T. E. and A. R. Dobell, 1962, "Random Number Generators," Society for Industrial and Applied Mathematics, Vol. 4, pp. 230-254.
6. Kahn, Hermann, 1956, "Applications of Monte Carlo," RM-1237-AEC, Santa Monica, CA: The Rand Corp., pp. 259.
7. Kellner, J., 1980, "Pascal Operand Formats or the Secret Life of a Variable," The Apple Orchard, Oct. 1980, pp. 38-40.
8. Knuth, D. E., 1981, "The Art of Computer Programming, Vol 2: Seminumerical Algorithms," Reading, MA: Addison-Wesley, pp. 688.
9. Levene, H. and J. Wolfowitz, 1944, "Asymptotic Distributions of Up and Down Runs," Annals Math. Statist., Vol. 15, pp. 58-69.
10. Owens, D. B., 1967, "Handbook of Statistical Tables," Reading, MA: Addison-Wesley, pp. 580.
11. Wald, A. and J. Wolfowitz, 1943, "An Exact Test for Randomness in the Non-Parametric Case Based on Serial Correlation," Annals of Math. Statist., Vol. 14, pp. 378-388.

SECTION 4

DISCUSSION OF RESULTS

The proposed random number generator has passed all the statistical tests to which it has been subjected. It is also very fast, approximately 80 times faster than the congruence multiplicative Pascal subroutine that it replaced. A multiplicative congruence method with a decently long period requires double precision multiplications, which is a slow process in microcomputers, when possible at all. The proposed assembly language subroutine uses only bit manipulations which involve shifts and only one Pascal addition, so it is implemented in a very efficient manner.

Of all the tests employed, the frequency and serial correlation tests are considered the "weakest," in the sense that most random number generators usually pass these tests. The runs test, as well as those tests that use nonparametric statistics, are considered the strongest. One may question the need for so many tests; this is, indeed, a valid question. In fact, it has been suggested (Knuth, 1981) that more computing time is usually spent testing random number generators than using them! However, the need for an exhaustive series of tests for random number generators that will be used in simulation work involving thousands and even millions of repetitions is well documented (Knuth, *ibid*). If one can prove that the program to be used is robust with respect to the quality (randomness) of the number stream and achieve this robustness with a fast generation, then the time spent in the testing was worthwhile.

Expected number of runs = 5000.5
 Expected standard deviation = 28.869
 Observed number of "up" runs = 4995
 Observed number of "down" runs = 5031
 $Z_{up} = (Exp - Obs)/Sdev = 0.1905$
 $Z_{down} = (Exp - Obs)/Sdev = 1.0565$
 Probability of chance occurrence for $Z_{up} = 0.8490$
 Probability of chance occurrence for $Z_{down} = 0.3908$.

The results for run lengths, using the modified runs technique were

<u>RUN</u>	<u>EXP UP</u>	<u>OBS UP</u>	<u>EXP DOWN</u>	<u>OBS DOWN</u>
1	1856.0	1903	1852.5	1892
2	1237.3	1216	1235.0	1230
3	464.0	443	463.1	420
4	123.7	127	123.5	131
5	25.8	18	25.7	25
6	5.2	5	5.1	7

Degrees of freedom = 6
 Chi-square Up runs = 4.9461
 Chi-square Down runs = 6.0224
 Probability of chance occurrence Up runs = 0.5507
 Probability of chance occurrence Down runs = 0.4207.

Hence, the random number generator proposed here passes the runs test handily.

RUNS TEST

A sequence of random numbers may be tested for the number and length of "runs up" and "runs down." In order to clarify the concept, consider the sequence of ten numbers

| 1 2 9 | | 8 | | 5 | | 3 6 7 | | 0 4 |

which displays a run "up" of length 3, followed by two runs of length 1, followed by another run of length 3, followed by a run of length 2. When estimating the probabilities of expected run lengths, Levene and Wolfowitz (1944) showed that a simple chi-square test cannot be applied to the discrepancies between observed and expected lengths since a long run is more likely to be followed by a short one, or, in other words, there is serial correlation between runs of different lengths. The exact test is horrendously complicated. However, Knuth (1981) has shown that if the number that breaks a run is simply thrown out, a chi-square test can be applied to the discrepancies. Since there is an ample supply of random numbers, we have chosen to implement the vastly simpler test that uses these modified runs, rather than the much more complex test that uses all the numbers. If the number that breaks a run is thrown out, the probability of a run of length R is

$$\text{Prob}\{\text{of run} = R\} = 1/R! - 1/(R+1)! .$$

Since this probability decreases very rapidly with R , the test was implemented using runs of one through five consecutive up (or down) numbers, and runs of length greater or equal to six were grouped together into a single category. The probability of a run of length greater than or equal to R is simply $1/R!$.

A preliminary test should consider the expected number of runs. The probability of the number of runs, when all numbers are used, is a normally distributed variate with parameters (for $N=10,000$),

$$\text{Mean} = E(R) = (N+1)/2 = 5000.5$$

$$\text{Var}(R) = 833.42 .$$

The results of the runs test for the number of runs (up and down), using all available random numbers were

$$\text{Mean} = E(R_h) = (S_1 - S_2)/(N-1)$$

$$\text{Var}(R_h) = (N-1)^{-1} [S_2 - S_4 + (N-2)^{-1} (S_1 - 4 S_1 S_2 + 4 S_1 S_3 + S_2 - 2 S_4) - (N-1)^{-1} (S_1 - S_2)^2] ,$$

where

$$S_k = \sum_{i=0}^{9972} u_i^k$$

is the kth power sum of the observations. Wald and Wolfowitz (ibid) have shown that R_h approaches the unit normal distribution for large N. Hence, the random variable

$$Z_h = [R_h - E(R_h)] [\text{Var}(R_h)]^{-1/2}$$

has a normal distribution with zero mean and unit standard deviation. This test is nonparametric in the sense that it does not depend on the assumption that the u_i 's are uniformly distributed. The results of this test were

Expected values of Corr Sums = 2467.97 .

<u>LAG</u>	<u>OBSERVED CORR SUMS</u>	<u>Z_h</u>	<u>PROB OF CHANCE OCCURRENCE</u>
1	2463.82	0.5044	0.6140
2	2463.78	0.5094	0.6105
3	2466.97	0.1215	0.9033
4	2459.60	1.0172	0.3090
5	2481.29	-1.6180	0.1057
6	2468.77	-0.0976	0.9222
7	2469.30	-0.1612	0.8719
8	2469.73	-0.2136	0.8308
9	2466.63	0.1631	0.8704
10	2459.93	0.9767	0.3287

Therefore, the proposed random number generator passes the serial correlation tests with up to 10 lags.

The variate

$$\chi^2 = \sum_{i=1}^5 (E_i - O_i)^2 / E_i$$

should be approximately distributed as a chi-square variate with 4 degrees of freedom. The results from this test were

<u>I</u>	<u>OBS</u>	<u>EXP</u>	<u>DIFF</u>
1	0	3.2	3.2
2	182	192.0	10.0
3	995	960.0	35.0
4	733	768.0	35.0
5	90	76.8	13.2

Degrees of freedom = 4

$$\chi^2 = 8.861$$

Probability of chance occurrence = 0.1816 .

SERIAL CORRELATION TEST

Consider a series u_0, u_1, \dots, u_{N-1} of random numbers. To check if this sequence is correlated in any way at equal intervals spaced h units apart, the serial correlation of the series can be obtained for various lags and tested against expectations. Serial correlation can be defined for the circular and noncircular cases (Bennet and Franklin, 1954). Only the noncircular test was applied. In the noncircular test, values $u_{i+h} \geq N$ are simply omitted, and only the serial correlation between the remaining pairs tested. Since N is very large, this is not a serious shortcoming. The test for serial correlation, originally developed by Wald and Wolfowitz (1943), requires that N be a prime number. The largest number available for the test is $N=9973$. Only lags from 1 through 10 were tested, since most of the important lags in simulation work are smaller than 10. Thus, the first 9973 numbers in the random sequence were used to compute the statistic

$$R_h = \sum_i u_i u_{i+h} \quad \begin{matrix} (h=1,2,\dots,10) \\ (i=0,1,\dots,9972) \end{matrix}$$

The mean and variance of the random variable R_h are, respectively, (Criegel and Shields, 1982),

"POKER" TEST

This test is conducted by dividing the N real numbers into $N/5$ groups of successive integer quintuples. Each five successive real numbers were converted into five integers using the scheme

1 if $0 < u_i \leq 0.2$
2 if $0.2 < u_i \leq 0.4$
3 if $0.4 < u_i \leq 0.6$
4 if $0.6 < u_i \leq 0.8$
5 if $0.8 < u_i \leq 1.0$.

Each quintuple was then classified into a "poker" hand,

Five different = all numbers different

Four different = one pair

Three different = two pairs, or three of a kind

Two different = full house, or four of a kind

One different = five of a kind.

The probability of each case can be derived using the Stirling numbers of the second kind (Crigler and Shield, 1982). The probabilities are

$$p_1 = 1/625 = 0.0016$$

$$p_2 = 12/125 = 0.096$$

$$p_3 = 12/25 = 0.48$$

$$p_4 = 48/125 = 0.384$$

$$p_5 = 24/625 = 0.0384$$

Note that the sum of all the probabilities is unity, as it should be.

Let

$E_i = N p_i$ = expected number of quintuples in the i th category

O_i = observed number of quintuples in the i th category.

Let

p_r probability of a gap of length r ($0 \leq r < t$)

p_t = probability of a gap of length t or greater than t .

Then

$$p_r = p(1 - p)^r \quad (0 \leq r \leq t-1)$$

$$p_r = (1 - p)^t \quad r = t .$$

If the stream is random, the variate

$$\chi^2 = \sum_{r=0}^t (Z_r - n p_t)^2 / (n p_t)$$

is distributed as a chi-square random variate with t degrees of freedom. For the test, the values

$$\alpha = 0.3$$

$$\beta = 0.6$$

$$t = 8$$

were selected following Crigler and Shields (1982). The results for the test were

<u>LENGTH</u>	<u>FREQ</u>	<u>EXPECTED</u>	<u>DIFF</u>
0	919	909.6	-9.4
1	627	636.7	9.7
2	472	445.7	-26.3
3	309	312.0	3.0
4	192	218.4	26.4
5	172	152.9	-19.1
6	95	107.0	12.0
7	72	74.9	2.9
8	174	174.8	0.8

Degrees of freedom = 8

$$\chi^2 = 8.8731$$

Probability of chance occurrence = 0.3531 .

is approximately distributed as a chi-square random variable with 5 degrees of freedom. For this particular test, $E_1 = 3333/6 = 555.5$.

The results of the test were

<u>TRIO</u>	<u>FREQ</u>	<u>DIFF</u>
(123)	552	3.5
(132)	556	-0.5
(213)	574	-18.5
(231)	544	11.5
(312)	564	-8.5
(321)	543	12.5

Degrees of freedom = 5

$$\chi^2 = 1.288$$

Probability of chance occurrence = 0.9361 .

GAP TEST

This test examines the lengths of "gaps" between numbers that fall into some prespecified range. A chi-square test, similar to the one used in the frequency test, is then used to check whether the differences between observed and expected number of occurrences in the preselected gaps could be expected by chance alone.

Let α and β be two numbers such that $0 \leq \alpha < \beta \leq 1$. Consider the sequence of random variates u_0, u_1, \dots, u_{N-1} as a cyclic sequence in which u_{N+j} is identified with u_j . Consider next the lengths of consecutive sequences $u_{j+0}, u_{j+1}, \dots, u_{j+r}$ in which u_{j+r} lies between α and β , but the other u values do not. Such a subsequence defines a gap of length r . If n of the N numbers u_0, u_1, \dots, u_{N-1} fall in the range $\alpha \leq u_j < \beta$, then there are n gaps in the cyclic sequence. Let

Z_r = number of gaps of length r , $0 \leq r < t$

Z_t = number of gaps of length t or greater

p = probability that $\alpha \leq u_j < \beta$.

Then

$$p = \beta - \alpha .$$

pair determines a location (bin) on a 10 X 10 matrix. If the pairs are random and uniformly distributed, each bin would be expected to fill with the same number of pairs of points. Let

$E_{i,j}$ = expected number of time the integer
pair i,j occurs = 100

$O_{i,j}$ = observed number of times the integer
pair i,j occurs.

Then the variate

$$\chi^2 = \sum_{i=0}^9 \sum_{j=0}^9 (E_{i,j} - O_{i,j})^2 / E_{i,j}$$

is approximately distributed as a chi-square variable with 99 degrees of freedom. The results from this test were

Degrees of freedom = 99

$\chi^2 = 88.0$

Probability of chance occurrence = 0.5088 .

The observed distribution of pairs is presented in appendix D.

PERMUTATION OF RANDOM TRIPLETS

In this test the random integer numbers obtained in the previous test are grouped into triplets according to a fixed rule. There are six different combinations for three numbers A, B, and C, when A, B, and C are ordered according to their absolute value: (A,B,C), (A,C,B), (B,A,C), (B,C,A), (C,A,B), and (C,B,A). Let

E_i = expected number of triplets in the i th category

O_i = observed number of triplets in the i th category

if each number is as likely to occur as any other, then the test variate

$$\chi^2 = \sum_{i=1}^6 (E_i - O_i)^2 / E_i$$

p_i = probability that an observation falls into the i th bin

E_i = expected number of observations in the i th bin

O_i = observed number of observations in the i th bin.

Then the variate

$$\chi^2 = \sum_{i=1}^{100} (E_i - O_i)^2 / E_i = \sum_{i=1}^{100} (100 - O_i)^2 / 100$$

is approximately distributed as a chi-square random variable with 99 degrees of freedom. The results of the test were

$$\begin{aligned} \text{Degrees of freedom} &= 99 \\ \chi^2 &= 115.86 \\ \text{Probability of chance occurrence} &= 0.1184 . \end{aligned}$$

The Kolmogorov-Smirnov test has the virtue that it is a distribution-free test. If the observed cumulative distribution for the selected 100 bins is calculated, the differences between the observed cumulative distribution and the expected theoretical distribution (the frequency at each successive bin should simply increase by $10,000/100 = 100$ variates) can be calculated. The Kolmogorov-Smirnov test estimates the probability of obtaining a discrepancy as large as the largest discrepancy found between expected and observed cumulative distributions (Owens, 1967). The results of this test were

$$\begin{aligned} \text{Max observed discrepancy} &= 0.0061 \\ \text{Probability of chance occurrence} &= 0.5249 . \end{aligned}$$

Therefore, the proposed stream passes both the frequency and the Kolmogorov-Smirnov test. The listing of observed frequencies is presented in appendix C.

DISTRIBUTION OF RANDOM PAIRS

This test is designed to determine if successive pairs of random numbers are uniformly and independently distributed. The sequence of N random numbers is converted into a sequence of random integers from 0 to 9, inclusive, by multiplying all the numbers by 10 and truncating the result. The resulting random integers are grouped into pairs, and each

moments, however, are particularly important so these should be the first tests applied to a stream. A unit random distribution has a theoretical mean of 0.5 and a variance of 1/12 (Knuth, 1981). For large values of N, the variates

$$z_m = (\text{Mean} - 0.5)(12 N)^{1/2}$$

$$z_{sd} = (\text{St Dev} - 0.2887)(24 N)^{1/2}$$

are normally distributed with zero mean and unit variance. Since positive and negative departures from the expected values are equally likely, two-tailed normal tests were applied to Z_m and Z_d . The values obtained were

$$\begin{aligned}\text{Expected Mean} &= 0.5 \\ \text{Observed Mean} &= 0.4977 \\ \text{Diff} &= 0.0023 \\ z_m &= 0.7978\end{aligned}$$

Probability of chance occurrence of $z_m = 0.4250$.

$$\begin{aligned}\text{Expected St Dev} &= 0.2887 \\ \text{Observed St Dev} &= 0.2875 \\ \text{Diff} &= 0.00113 \\ z_{sd} &= 0.5542\end{aligned}$$

Probability of chance occurrence of $z_{sd} = 0.5795$.

The proposed generator passes the mean and standard deviation tests.

FREQUENCY AND KOLMOGOROV-SMIRNOV TEST

These two tests examine the expected frequency distribution of the random numbers by sorting them into frequency bins. These two tests are usually tabulated together since it is convenient for the formatting of the results. The frequency test is probably the test most commonly applied and one that most random number generators pass handily. The unit range between zero and unity was divided into 100 equally spaced frequencies, and the generated random numbers were distributed into these 100 frequency bins. Let

have been produced by a perfect random number generator, the fact that the samples are of finite rather than infinite size makes it unlikely that the test expectations will be met perfectly. In fact, it would be a most unusual occurrence if they did. Suppose, however, that for each test it is possible to calculate a certain test statistic "z" which has a known theoretical probability distribution for a perfectly random stream of numbers. If the probability of occurrence of the "z" calculated from the sampled stream under test is reasonably high, one can be confident that the number generator used is also reasonably random, as far as this particular test is concerned. What might be considered "reasonably high"? The answer depends on the application intended for the random numbers. A widely accepted criterion in simulation work is that a probability of five percent or less for the occurrence of "z" is too low. Therefore, in what follows, the result of a test will be considered acceptable if the probability of occurrence of the test statistic that accompanies the test exceeds 0.05. This is equivalent to accepting that the observed discrepancies between actual and expected values in the test were due to unavoidable chance fluctuations due to sampling.

The most commonly used test statistics have either a normal or a chi square (χ^2) distribution and their probabilities of occurrence can be readily computed. As an example, the five percent limit for a unit normal distribution (zero mean and unit standard deviation) is $z = \pm 1.96$.

Therefore, in all the tests that follow, the expression "probability of chance occurrence" is equivalent to the probability that the observed stream could have occurred at random from the output of a perfect random number generator. If this probability is greater than five percent, we are willing to accept the stream. One must keep in mind, however, that at this acceptance level, five percent of the streams from even a perfect random number generator would have been rejected. For this reason, it is unwise to reject a random number generator for an occasional failure to pass a test at a specified significance level.

The various tests that were applied will be discussed only briefly. For those interested in the details, Knuth (1981) and Crigler and Shields (1982) can be consulted. The Pascal program used in the testing is listed in appendix A so that other users may have access to it. All the tests were performed using $N=10,000$ random deviates unless stated otherwise.

MEAN AND STANDARD DEVIATION

These tests (actually two separate tests that "go" together) are applied as a check on the correctness of the expected moments of the distribution rather than to test its randomness. The first and second

```

.PUBLIC RNB,SEED,MASK
LDX #7                ;# OF BITS TO BE SHIFTED INTO RNB+2
LDA #0                ;CLEAR ACCUMULATOR
STA TEMP              ;CLEAR STORAGE LOCATIONS
STA RNB
STA RNB+1
LOOP 1 JSR GETCBIT      ;GO TO GETCBIT
      ROL TEMP          ;PUT CARRY BIT IN LSB OF TEMP
      DEX              ;DECREMENT X
      BNE LOOP1         ;SHIFT 7 BITS INTO TEMP
      LDA RNB+2
      AND #080          ;SAVE EXPONENT BIT IN RNB+2
      ORA TEMP          ;COMBINE IT WITH TEMP
      STA RNB+2         ;STORE IN RNB+2
      LDX #8            ;# OF BITS TO SHIFT INTO RNB+1
LOOP 2 JSR GETCBIT      ;STORE 8 BITS IN RNB+1
      ROL RNB+1
      DEX
      BNE LOOP2
      LDX #8            ;RESET COUNTER FOR RNB
LOOP 3 JSR GETCBIT      ;STORE 8 BITS IN RNB
      ROL RNB
      DEX
      BNE LOOP3
      RTS              ;RETURN TO CALLING PROGRAM
;
;
GETCBIT LDA SEED+3      ;TEST LEFTMOST BIT OF SEED
      BPL SHFT          ;IF 0 GO TO SHFT
      LDY #3            ;ELSE
EORLOOP LDA SEED,Y
      EOR MASK,Y        ;EOR SEED WITH MASK AND
      STA SEED,Y        ;STORE THE RESULTS BACK IN SEED
      DEY              ;REPEAT FOR EACH OF 4 BYTES OF
      BPL EORLOOP       ;MASK AND SEED
      JSR SHSEED        ;GO TO SHSEED
      SEC              ;SET CARRY BIT TO 1
      RTS              ;RETURN TO LOOPX
SHFT JSR SHSEED
      CLC              ;SET CARRY BIT TO 0
      RTS
;
;
SHSEED ASL SEED          ;SHIFT 4 BYTES OF SEED LEFT ONE BIT
      ROL SEED+1
      ROL SEED+2
      ROL SEED+3
      RTS              ;RETURN
TEMP .BYTE              ;BYTE OF TEMPORARY STORAGE
      .END

```

APPENDIX B
LISTINGS OF TEST PROGRAMS

```
(*$$+*)
PROGRAM TESTR; (*28 FEB 1983 ALL TESTS EXCEPT RUNS TEST*)

USES TRANSCEND;

NFREQ,NF,NT,NGAPS,CNT,N,PAIR,TRIO,POK,I,TJ,PDIF,F,
ROW,COL,MAXPRIME,TPAIR,TTRIO,TPOK,TFREQ,TSER:INTEGER;
X,S1,S2,S3,S4,RNB,Z1,Z2                :REAL;
NUMPOK,SPOK                            :ARRAY[1..5] OF INTEGER;
SFREQ                                  :ARRAY[0..99] OF INTEGER;
SSER                                  :ARRAY[1..10] OF REAL;
PERM                                  :ARRAY[1..3] OF REAL;
SPERM                                :ARRAY[1..6] OF INTEGER;
XSER                                  :ARRAY[0..10] OF REAL;
SGAP                                  :ARRAY[0..8] OF INTEGER;
SPAIRS                                :ARRAY[0..9,0..9] OF INTEGER;
SEED,MASK                             :PACKED ARRAYS[0..3] OF 0..255;
PR                                    :TEXT;

(****)
PROCEDURE RNUM; (*GENERATES RND, A RANDOM UNIT VARIATE*)
EXTERNAL;

(****)
FUNCTION RND:REAL;
BEGIN
  RNUM;
  RND:=RNB-1.0
END;

(****)
FUNCTION CNORMAL (Y:REAL):REAL; (*COMPLEMENT OF THE NORMAL PROB*)
VAR T,Q:REAL;

BEGIN
  T:=1/(1+0.33267*ABS(Y));
  Q:= 0.3989422*T*((0.937298*T-0.1201676)*T+0.4361836);
  Q:=LN(Q)-0.5*Y*Y;
  IF Q>-87 THEN Q:=EXP(Q) ELSE Q:=0;
  IF Y>0 THEN CNORMAL:=Q ELSE CNORMAL:=1-Q
END;
```

```

(****)
FUNCTION PROB(G,Z:REAL):REAL; (*PROB OF CHI-SQUARE*)
VAR N,I:INTEGER;
    S,T,X,Y:REAL;

BEGIN
  IF G<30 THEN
    BEGIN
      N:=TRUNC(G);
      T:=1;
      IF G-N>0 THEN
        BEGIN
          I:=0;
          S:=0;
          X:=Z+Z;
          Y:=SQRT(X) ;
          IF N>0 THEN
            BEGIN
              WHILE I<N DO
                BEGIN
                  I:=I+1;
                  S:=S+T;
                  T:=T*X/(I+I+1);
                END;
              S:=LN(S*Y)-Z-0.225791;
              IF S>-87 THEN S:=EXP(S) ELSE S:=0;
            END;
            PROB:=2*CNORMAL (Y)+S;
          END
        ELSE
          BEGIN
            S:=-1;
            I:=1;
            WHILE I<N DO
              BEGIN
                T:=T*Z/I;
                S:=S+T;
                I:=I+1;
              END;
            S:=LN(S)-Z;
            IF S>-87 THEN PROB:=EXP(S) ELSE PROB:=0;
          END;
        END
      ELSE
        BEGIN
          T:=9*G;
          X:=(EXP(LN(Z/G)/3)-1+1/T)*SQRT(T);
          X:=X+(((0.009191*X-0.004772)*X-0.026868)*X+0.00445)/G;
        END
      END
    END
  END

```

```

        PROB:=CNORMAL(X);
    END

END;

(****)
PROCEDURE INITIAL; (*SETS INITIAL PARAMETERS*)
VAR I,J,N:INTEGER;

BEGIN
    RNB:=1.0;
    SEED[0]:=97;
    SEED[1]:=101;
    SEED[2]:=103;
    SEED[3]:=107;
    MASK[0]:=83;
    MASK[1]:=181;
    MASK[2]:=118;
    MASK[3]:=0;
    NGAPS:=0;
    CNT:=0;
    PAIR:=1;
    TRIO:=1;
    POK:=1;
    S1:=0;
    S2:=0;
    S3:=0;
    S4:=0;
    Z1:=0;
    Z2:=0;
    N:=NFREQ-1;
    FOR I:=0 TO N DO SFREQ[I]:=0;
    FOR I:=1 TO 5 DO SPOK[I]:=0;
    FOR I:=1 TO 10 DO SSER[I]:=0;
    FOR I:=1 TO 6 DO SPERM[I]:=0;
    FOR I:=0 TO 8 DO SGAP[I]:=0;
    FOR I:=0 TO 9 DO FOR J:=0 TO 9 DO SPAIRS [I,J]:=0
END;

(****)
PROCEDURE MOMENTS(J:INTEGER;Y:REAL); (*COMPUTES THE FIRST 4 MOMENTS *)
VAR Z:REAL;

BEGIN
    Z1:=Z1+Y;
    Z:=Y*Y;
    Z2:=Z2+Z;
    IF J<MAXPRIME THEN

```



```

      BEGIN
        S1:=Z1;
        S2:=Z2;
        Z:=Z*Y;
        S3:=S3+Z;
        S4:=S4+Z*Y;
      END
    END;

  (****)
  PROCEDURE SERIAL(J:INTEGER;Y:REAL); (*OBTAINS SERIAL CORRELATION SUMS*)
  VAR I:INTEGER;
      Z:REAL;

  BEGIN
    IF J<10 THEN XSER[J]:=Y ELSE
      BEGIN
        XSER[10]:=Y;
        Z:=XSER[0];
        FOR I:=1 TO 10 DO
          BEGIN
            SSER[I]:=SSER[I]+Z*XSER[I];
            XSER[I-1]:=XSER[I];
          END;
        END;
      END
    END;

  (****)
  PROCEDURE GAPS(Y:REAL);
  (*COUNTS HOW MANY CONSEC VARIATES FALL BETWEEN 0.3 AND 0.6*)

  BEGIN
    IF (Y<0.3) OR (Y>0.6) THEN CNT:=CNT+1
    ELSE
      BEGIN
        NGAPS:=NGAPS+1; (*UPDATE NUMBER OF GAPS*)
        IF CNT>7 THEN CNT:=8;
        SGAP[CNT]:=SGAP[CNT]+1; (*UPDATE SUMS*)
        CNT:=0;
      END
    END;

  (****)
  PROCEDURE POKER(P:INTEGER); (*COUNTS VARIOUS POSSIBLE POKER HANDS*)
  VAR J,K,M,L :INTEGER;

  BEGIN
    NUMPOK[POK]:=P;

```

```

POK:=POK+1;
IF POK=6 THEN
  BEGIN
    PDIF:=1;    (*NUMBER OF DIFF DIGITS*)
    J:=2;
    REPEAT
      K:=NUMPOK[J];
      M:=1;
      WHILE (M<J) AND (NUMPOK[M]<>K) DO M:=M+1;
      IF M=J THEN PDIF:=PDIF+1;
      J:=J+1;
    UNTIL J>5;
    SPOK[PDIF]:=SPOK[PDIF]+1;  (*UPDATE POKER SUMS*)
    POK:=1;
  END
END;

(****)
PROCEDURE HEADING; (*PRINTS HEADINGS*)
VAR I:INTEGER;

BEGIN
  WRITE(PR,CHR(ORD(12))); (*ADVANCE ONE PAGE)
  FOR I:=1 TO 3 DO WRITELN(PR);
  WRITE(PR,CHR(ORD(14))); (*DOUBLE WIDTH*)
  WRITELN(PR,' TESTING A UNIT RANDOM NUMBER GENERATOR');
  WRITELN(PR);
  WRITELN(PR,'THERE ARE ',NT:5,' RANDOM VARIATES AVAILABLE');
  WRITELN(PR)
END;

(****)
PROCEDURE UNDRLNON; (*UNDERLINE MODE ON *)
BEGIN
  WRITE(PR,CHR(ORD(27)),CHR(ORD(45)),CHR(ORD(1)))
END;

(****)
PROCEDURE UNDRLN OFF; (*UNDERLINE MODE OFF*)
BEGIN
  WRITE(PR,CHR(ORD(27)),CHR(ORD(45)),CHR(ORD(0)))
END;

****)
PROCEDURE MNSDTEST; (*TESTS MEAN, SDEV OF DISTRIBUTION AND KOLMOGOROFF TEST*)
VAR  I:INTEGER;
      MN,SD,TM,TD:REAL;

```

```

BEGIN
  HEADING;
  UNDRNLN;
  WRITELN(PR,'TEST OF OBSERVED MEAN AND STD DEV OF DISTRIBUTION');
  UNDRLNOFF;
  FOR I:=1 TO 3 DO WRITELN(PR);
  WRITELN(PR,'N=',NT:5,' VARIATES USED');
  WRITELN(PR);
  WRITELN(PR);
  MN:=Z1/NT;
  SD:=SQRT(Z2/NT-MN*MN);
  TM:=(0.5-MN)*SQRT(12.0*NT);
  TD:=(0.288675-SD)*SQRT(24.0*NT);
  WRITELN(PR,'EXPCT MEAN=0.5000');
  WRITELN(PR,'OBSVD MEAN=',MN:6:4);
  WRITELN(PR,'      DIFF=',(0.5-MN):6:5);
  WRITELN(PR,'      T=',TM:6:4);
  WRITELN(PR,'PROB OF CHANCE OCCURRENCE=',2*CNORMAL(ABS(TM)):6:4);
  FOR I:=1 TO 3 DO WRITELN(PR);
  WRITELN(PR,'EXPECT SDEV=0.2887');
  WRITELN(PR,'OBSVD SDEV=',SD:6:4);
  WRITELN(PR,'      DIFF=',(0.288675-SD):6:5);
  WRITELN(PR,'      T=',TD:6:4);
  WRITELN(PR,'PROB OF CHANCE OCCURRENCE=',2*CNORMAL(ABS(TD)):6:4);
END;

(****)
PROCEDURE FREQKOLTEST; (*PERFORMS A FREQCY CHISQ AND KOLM-SMRNOV TEST*)
VAR I,LO,HI      :INTEGER;
    SUMFRQ,CUM,EXPFRQ,DIF,MAX,CHI,EXPCUM,KOL,PRB      :REAL;

```

```

BEGIN
  CHI:=0;
  SUMFRQ:=0;
  MAX:=0;
  EXPFRQ:=NT/NFREQ;
  EXPCUM:=0;
  LO:=0;
  REPEAT
    HI:=NFREQ-1;
    IF HI-LO>33 THEN HI:=LO+33; (*BREAKS TABLE INTO READABLE FORMAT*)
    HEADING;
    UNDRNLN;
    WRITELN(PR,'FREQUENCY DISTRIBUTION AND KOLMOGOROFF-SMIRNOV TEST');
    UNDRLNOFF;
    FOR I:=1 TO 3 DO WRITELN(PR);
    WRITELN(PR,NT:5,' VARIATES ARE DISTRIBUTED INTO',NFREQ:4,' BINS.EXPECTD');
    WRITELN(PR,' FREQUENCY IS ',EXPFRQ:5:1,' ."DIF"=EXPCTD-ACTUAL FREQUENCY');

```

```

WRITELN(PR,"KOL" IS THE DIFF BETWEEN EXPCTD AND ACTUAL CUMUL FREQUENCY");
WRITELN(PR);
WRITELN(PR,"  BIN  OBS  DIF  CUM  KOL");
FOR I:=LO TO HI DO
  BEGIN
    DIF:=EXPFRQ-SFREQ[I];
    CHI:=CHI+DIF*DIF;
    SUMFRQ:=SUMFRQ+SFREQ[I];
    EXPCUM:=EXPCUM+EXPFRQ;
    CUM:=SUMFRQ/NT;
    KOL:=(EXPCUM-SUMFRQ)/NT;
    IF ABS(KOL)>MAX THEN MAX:=ABS(KOL);
    WRITELN(PR,I:5,SFREQ[I]:5,DIF:6:1,CUM:8:4,KOL:8:4);
  END;
LO:=HI+1;
UNTIL LO=NFREQ;
FOR I:=1 TO 2 DO WRITELN(PR);
CHI:=CHI/EXPFRQ;
WRITELN(PR,"DEGREES OF FREEDOM=",NFREQ-1:3,"      CHISQUARE=",CHI:8:2);
WRITELN(PR,"PROB OF CHANCE OCCURRENCE=",PROB((NFREQ-1)/2.0,CHI/2.0:5:4);
FOR I:=1 TO 2 DO WRITELN(PR);
WRITELN(PR,"MAX KOLMOGOROV-SMIRNOV DIFF=",MAX:6:4);
PRB:=-2.0*MAX*MAX*NT;
IF PRB>-87 THEN PRB:=1-EXP(PRB) ELSE PRB:=1.0;
WRITELN(PR,"PROBABILITY OF CHANCE OCCURRENCE=",PRB:5:4)
END;

(****)
PROCEDURE GAPTEST;
VAR  I:INTEGER;
    CHI,P,Q,T,EXPCT,DIF,PRB:REAL;

BEGIN
  HEADING;
  CHI:=0;
  P:=0.3;
  Q:=0.7
  T:=1.0;
  UNDRNLNON;
  WRITELN(PR,"GAP TEST");
  UNDRLNOFF;
  FOR I:=1 TO 3 DO WRITELN(PR);
  WRITELN(PR,"THE NUMBER OF CONSECUTIVE RND VARIATES THAT FALL OUTSIDE;);
  WRITELN(PR,"THE RANGE (0.3-0.6) FORM THE "GAPS".  GAPS LONGER THAN 7 FORM");
  WRITELN(PR,"THE CATEGORY OF >7: OR=8. THE CUMULATIVE SUMS OF GAP RUNS FROM");
  WRITELN(PR,"0 THRU 8 ARE UPDATED AND THEIR ACTUAL FREQ IS COMPARED WITH");
  WRITELN(PR,"THE THEORETICALLY EXPECTED FREQUENCY FOR A CHISQUARE TEST");
  WRITELN(PR);

```

```

WRITELN(PR,'LTH FREQ  XPCTD DIFF');
FOR I:=0 TO 8 DO
  BEGIN
    IF I<8 THEN PRB:=P*T ELSE PRB:=T;
    EXPCT:=PRB*NGAPS;
    DIF:=EXPCT-SGAP[I];
    CHI:=CHI+DIF*DIF/EXPCT;
    T:=T*Q;
    WRITELN(PR,I:4,SGAP[I]:6,EXPCT:6:1,DIF:6:1);
  END;
WRITELN(PR);
WRITELN(PR);
WRITELN(PR,'DEGREES OF FREEDOM=8  CHISQUARE=',CHI:6:5);
WRITELN(PR,'PROBABILITY OF CHANCE OCCURRENCE=',PROB(4,CHI/2.0):6:4)
END;

(****)
PROCEDURE POKERTEST;
VAR  I,N:INTEGER;
    CHI,DIF :REAL;
    EXPC:ARRAY[1..5] OF REAL;

BEGIN
  HEADING;
  CHI:=0;
  N:= NT DIV 5;
  EXPC[1]:= 0.0016*N;
  EXPC[2]:=0.096*N;
  EXPC[3]:=0.48*N;
  EXPC[4]:=0.384*N;
  EXPC[5]:=0.0384*N;
  UNDRNLNON;
  WRITELN(PR,'"POKER" TEST');
  UNDRLNOFF;
  FOR I:=1 TO 3 DO WRITELN(PR);
  WRITELN(PR,'A TOTAL OF ',N:4,' HANDS OF 5 RND DIGITS EACH ARE FORMED AND');
  WRITELN(PR,'TESTED FOR THE EXPCTD FREQ OF OCCURENCE OF 1 THRU 5 DIFF DIG');
  WRITELN(PR);
  WRITELN(PR,' I FREQ  EXPC  DIFF');
  FOR I:=1 TO 5 DO
    BEGIN
      DIF:=EXPC[I]-SPOK[I];
      CHI:=CHI+DIF*DIF/EXPC[I];
      WRITELN(PR,I:2,SPOK[I]:6,EXPC[I]:6:1,DIF:6:1);
    END;
  WRITELN(PR);
  WRITELN(PR,'DEGREES OF FREEDOM=4  CHISQUARE=',CHI:6:3);
  WRITELN(PR,'PROBABILITY OF CHANCE OCCURRENCE=',PROB(3,CHI/2.0):6:4)

```

END;

(****)

PROCEDURE PAIRS(L:INTEGER); (*OBTAINS RANDOM PAIRS*)

BEGIN

IF PAIR<0 THEN

BEGIN

COL:=L;

SPAIRS[ROW,COL]:=SPAIRS[ROW,COL]+1;

END

ELSE ROW:=L;

PAIR:=-PAIR

END;

(****)

PROCEDURE TRIOS(Y:REAL); (*OBTAINS RANDOM TRIOS*)

R MI,MA,L,J :INTEGER;

BEGIN

PERM[TRIO]:=Y;

TRIO:=TRIO+1;

IF TRIO=4 THEN

BEGIN

IF (PERM[1]<PERM[2]) AND (PERM[1]<PERM[3]) THEN

BEGIN

MI:=1;

IF PERM[2]>PERM[3] THEN MA:=2 ELSE MA:=3;

END

ELSE

BEGIN

IF (PERM[1]>PERM[2]) AND (PERM[1]>PERM[3]) THEN

BEGIN

MA:=1;

IF PERM[2]<PERM[3] THEN MI:=2 ELSE MI:=3;

END

ELSE

BEGIN

IF (PERM[1]>PERM[2]) AND (PERM[1]<PERM[3]) THEN

BEGIN

MI:=2;

MA:=3;

END

ELSE

BEGIN

MI:=3;

MA:=2;

```

        END;
    END;
END;
CASE MI OF
    1: IF MA=2 THEN TJ:=2 ELSE TJ:=1;
    2: IF MA=3 THEN TJ:=3 ELSE TJ:=5;
    3: IF MA=1 THEN TJ:=6 ELSE TJ:=4;
END;
SPERM[TJ]:=SPERM[TJ]+1;
TRIO:=1;
END
END;

(****)
PROCEDURE TRIOTEST;
VAR I,J:INTEGER;
    CHI,DIF,EXPCT:REAL;
BEGIN
    HEADING;
    UNDRNLN;
    WRITELN(PR,'TEST FOR FREQ OF OCCURRENCE OF RANDOM PERMUT OF DIGIT TRIPLETS');
    UNDRLNOFF;
    WRITELN(PR);
    CHI:=0;
    EXPCT:=TTRIO/18.0;
    J:=TRUNC(EXPCT*6.0);
    WRITELN(PR,'THERE ARE 6 DIFF PERMUTNS OF DIGITS IN A TRIPLET. A TOTAL OF');
    WRITELN(PR,'J:4,' TRIOS ARE FORMD, EACH PERM WITH EXPCTD FREQ OF',EXPCT:5:1);
    WRITELN(PR,'"DIF" IS THE DIFFERENCE BETWEEN EXPCTD AND OBSERVD FREQUENCIES');
    WRITELN(PR);
    WRITELN(PR,' TRIO FREQ DIFF');
    FOR I:=1 TO 6 DO
        BEGIN
            CASE I OF
                1:WRITE(PR,'(123)');
                2:WRITE(PR,'(132)');
                3:WRITE(PR,'(213)');
                4:WRITE(PR,'(231)');
                5:WRITE(PR,'(312)');
                6:WRITE(PR,'(321)');
            END;
            DIF:=EXPCT-SPERM[I];
            CHI:=CHI+DIF*DIF;
            WRITELN(PR,SPERM[I]:6,DIF:6:1);
        END;
    WRITELN(PR);
    WRITELN(PR);

```

```

    CHI:=CHI/EXPCT;
    Writeln(PR,'DEGREES OF FREEDOM=5    CHISQUARE=',CHI:8:4);'
    Writeln(PR,'PROBABILITY OF CHANCE OCCURRENCE=',PROB92.5,CHI/2.0):6:4)
END;

(****)
PROCEDURE PAIRTEST;
VAR I,J,N1:INTEGER;
    CHI,EXPCT,DIF:REAL;

BEGIN
    HEADING;
    CHI:=0;
    N1:=TRUNC(NT/2.0);
    EXPCT:=N1/100.0;
    UNDRNLN;
    Writeln(PR,'TEST FOR DISTRIBUTION OF RANDOM PAIRS');
    UNDRLNOFF;
    FOR I:=1 TO 3 DO Writeln(PR);
    Writeln(PR,'A TOTAL OF ', N1:5,' PAIRS OF 10 RANDOM DIGITS (0 THRU 9)
        ARE');
    Writeln(PR,'FORMED AND DISTRIBUTED INTO 100 FREQUENCY BINS. THE EXPCTD');
    Writeln(PR,'NUMBER OF PAIRS IN EACH BIN IS ',EXPCT:4:1,' . THE TEST');
    Writeln(PR,'USES THE CHI SQ STAT TO CALC THE PROB OF CHANCE OCCURRENCE');
    Writeln(PR,'OF THE OBSERVED FREQUENCY DISTRIBUTION');
    Writeln(PR);
    Writeln(PR);
    Writeln(PR,'
                                BINS');
    WRITE(PR,'
    ');
    FOR I:=0 TO 9 DO WRITE(PR,I:5);
    FOR I:=1 TO 2 DO Writeln(PR);
    FOR I:=0 TO 9 DO
        BEGIN
            WRITE(PR,I:5);
            FOR J:=0 TO 9 DO
                BEGIN
                    WRITE(PR,SPAIRS[I,J]:5);
                    DIF:= EXPCT-SPAIRS[I,J]
                    CHI:=CHI+DIF*DIF;
                END;
            Writeln(PR);
        END;
    FOR I:=1 TO 3 DO Writeln(PR);
    CHI:=CHI/EXPCT;
    Writeln(PR,'DEGREES OF FREEDOM=99    CHISQUARE='CHI:4:1);
    Writeln(PR,'PROBABILITY OF CHANCE OCCURRENCE='PROB(44.5,CHI/2):6:4);

```



```

(****)
PROCEDURE SERIALTEST;
VAR N,N1,I:INTEGER;
    A,B,C,D,EXPCT,SD,T,PRB:REAL;

BEGIN
    HEADING;
    N:=MAXPRIME-1;
    N1:=MAXPRIME-2;
    A:=S1*S1;
    B:=A*A;
    C:=S2*S2;
    D:=A-S2;
    D:=D*D;
    EXPCT:=(A-S2)/N;
    SD:=SQRT((C-S4+(B+4*S1*(S3-S1*S2)+C-S4-S4)/(N1-D/N)/N);
    UNDRNLN;
    WRITELN(PR,'SERIAL CORRELATION TEST FOR LAGS 1 THRU 10');
    UNDRLNOFF;
    FOR I:=1 TO 3 DO WRITELN(PR);
    WRITELN(PR,'NON-CIRCULAR SERIAL CORR BETWEEN THE FIRST ',N+1:5,' VARIATES');
    WRITELN(PR);
    WRITELN(PR,'THE OBSERVED MOMENTS ARE');
    WRITELN(PR,'S1='S1:9:4,'          S2=',S2:9:4);
    WRITELN(PR,'S3='S3:9:4,'          S4=',S4:9:4);
    FOR I:=1 TO 3 DO WRITELN(PR);
    WRITELN(PR,'THE EXPECTED VALUE OF CORR SUMS IS ',EXPCT:6:4);
    WRITELN(PR,'THE EXPECTED ST DEV OF CORR SUMS IS ',SD:6:4);
    WRITELN(PR);
    WRITELN(PR,"LAG CORRSUM      T      PROB");
    FOR I:=1 TO 10 DO
        BEGIN
            T:=(EXPCT-SSER[1])/SD;
            PRB:=2*CNORMAL(ABS(T));
            WRITELN(PR,I:3,SSER[1]:9:4,T:8:4,PRB:8:4);
        END
    END;

(*****)
(**MAIN PROGRAM**)
(*****)

BEGIN
    REWRITE(PR,'PRINTER:'); (*ENABLE PRINTER*)
    WRITE('HOW MANY TRIALS? ');
    READLN(NT);
    N:=NT-1;
    WRITE('HOW MANY BINS FOR FREQUENCY TEST (NFREQ<101)? ');
    READL(NFREQ);

```

```

WRITE('MAX PRIME NUMBER FOR SERIAL TEST?(MAXPRIME<NTRIALS-9) ');
READLN(MAXPRIME);
INITIAL;
TPAIR:=2*TRUNC(NT/2.0);
TTRIO:=3*TRUNC(NT/3.0);
TPOK:=5*TRUNC(NT/5.0);
TFREQ:=NFREQ*TRUNC(NT/NFREQ);
TSER:=MAXPRIME+10;
FOR I:=0 TO N DO
  BEGIN
    X:=RND;
    WRITELN(I); (*SCREEN VISIBLE REMINDER*)
    MOMENTS (I,X); (*1ST THRU 4TH MOMENTS*)
    GAPS(X);(*COUNT OCCURRCES OF 0.3<X<0.6*)
    JF:=TRUNC(NFREQ*X);
    IF I<TFREQ THEN SFREQ[JF]:=SFREQ[JF]+1;(*DISTR VARIATES INTO NFREQ BINS*)
    IF I<TPAIR THEN PAIRS(TRUNC(10.0*X)); (*DISTR DIGIT PAIRS INTO 100 BINS*)
    IF I<TTRIO THEN TRIOS(X);(DISTRB TRIPLETS INTO 6 CATEGORIES*)
    IF I<TPOK THEN POKER(TRUNC(5.0*X));(*DISTRB POKER HANDS INTO 5 CATEG*)
    IF I<TSER THEN SERIAL(I,X); (*FORM SERIAL CORR PRODS, LAGS 1 TO 10*)
  END;
MNSDTEST; (*MEAN AND SDEV TESTS*)
FREQKOLTEST; (*CHISQ FREQ DISTR AND KOLMOGOROFF-SMIRNOV TESTS*)
PAIRTEST; (*TEST DISTR OF RANDOM PAIRS*)
TRIOTEST; (*TEST DISTR OF RANDOM PERMUTATIONS OF TRIOS*)
GAPTEST; (*TEST GAPS FOR VARIATES THAT FALL OUTSIDE 0.3<X<0.6*)
POKERTEST; (*TEST DISTR OF POKER HANDS (5 RANDOM DIGITS)*)
SERIALTEST; (*SERIAL CORR TEST FOR LAGS 1 THRU 10*)
FOR I:=1 TO 2 DO WRITE(PR,CHR(ORD)12)))
END.

```

```

B(*$S+*)
PROGRAM TESTRUNS: (*2 MAR 1983 TESTS MODIFIED RUNS UP AND DOWN
AND TOTAL NUMBER OF ORDINARY RUNS*)

USES TRANSCEND;

TYPE SUMCASES=ARRAY[1..6] OF INTEGER;

VAR
NT,I,IJ,STUP,STDW                                :INTEGER;
RND,XR,LOUPOR,LODWOR,LOUPMD,LODWMD                :REAL;
CHIUP,CHIDW,PRB                                    :REAL;
CTUPOR,CTUPMD,CTDWOR,CTDWMD                        :INTEGER;
SEED,MASK                                           :PACKED ARRAY[0..3] OF 0..255;
SUPOR,SUPMD,SDWOR,SDWMD                           :ARRAY[1..6] OF INTEGER;
PR                                                  :TEXT;

(****)
PROCEDURE RNUM;
EXTERNAL;

(****)
FUNCTION RND:REAL;
BEGIN
  RNUM;
  RND:=RNB-1.0
END;

(****)
FUNCTION CNORMAL(Y:REAL):REAL; (*COMPLEMENT OF NORMAL PROB*)
VAR T,Q:REAL;

BEGIN
  T:=1/(1+0.33267*ABS(Y));
  Q:=0.3989422*T*((0.937298*T-0.1201676)*T+0.4361836);
  Q:=LN(Q)-0.5*Y*Y;
  IF Q>-87 THEN Q:=EXP(Q) ELSE Q:0;
  IF Y>0 THEN CNORMAL:=Q ELSE CNORMAL:=1-Q
END;

(****)
FUNCTION PROB(G,Z:REAL):REAL; (*PROB OF CHI-SQUARE*)
VAR N,I:INTEGER;
    S,T,X,Y:REAL;

BEGIN

  IF G<30 THEN
    BEGIN

```

```

N:=TRUNC(G);
T:=1;
IF G-N>0 THEN
  BEGIN
    I:=0;
    S:=0;
    X:=Z+Z;
    Y:=SQRT(X);
    IF N>0 THEN
      BEGIN
        WHILE I<N DO
          BEGIN
            I:=I+1;
            S:=S+T;
            T:=T*X/(I+I+1);
          END;
          S:=LN(S*Y)-Z-0.225791;
          IF S>-87 THEN S:=EXP(S) ELSE S:=0;
        END; (*OF WHILE*)
        PROB:=2*CNORMAL(Y)+S;
      END
    ELSE
      BEGIN
        S:=1;
        I:=1;
        WHILE I<N DO
          BEGIN
            T:=T*Z/I;
            S:=S+T;
            I:=I+1;
          END; (*OF WHILE*)
          S:=LN(S)-Z;
          IF S>-87 THEN PROB:=EXP(S) ELSE PROB:=0;
        END;
      END
    ELSE
      BEGIN
        T:=9*G;
        X:=(EXP(LN(Z/G)/3)-1+1/T)*SQRT(T);
        X:=X+(((0.009191*X-0.004772)*X-0.026868)*X+0.00445)/G;
        PROB:=CNORMAL(X);
      END
    END;
  END;

(****)
PROCEDURE UNDRNLN; (*UNDERLINE MODE ON*)

BEGIN

```

```

WRITE(PR,CHR(ORD(27)),CHR(ORD(45)),CHR(ORD(1)))
END;

```

```

(****)
PROCEDURE UNDRLOFF; (*UNDERLINE MODE OFF*)

```

```

BEGIN
WRITE(PR,CHR(ORD(27)),CHR(ORD(45)),CHR(ORD(0)))
END;

```

```

(****)
PROCEDURE HEADING;
VAR I:INTEGER;

```

```

BEGIN
WRITE(PR,CHR(ORD(12))); (*PAGE*)
FOR I:=1 TO 3 DO WRITELN(PR);
WRITE(PR,CHR(ORD(14)));
WRITELN(PR,` TESTING A UNIT RANDOM NUMBER GENERATOR`);
WRITELN(PR);
WRITELN(PR,`THERE ARE `NT:5,` RANDOM VARIATES AVAILABLE`);
WRITELN(PR);
UNDRLOFF;
WRITELN(PR,`RUNS TEST`);
UNDRLOFF;
FOR I:=1 TO 3 DO WRITELN(PR);
WRITELN(PR,`A RUN CONSISTS OF R CONSECUTIVE VARIATES EACH LARGER`);
WRITELN(PR,`OR SMALLER THAN THE PREVIOUS ONE. FOR ORDINARY (ORD) RUNS`);
WRITELN(PR,`ALL RNUMBERS ARE USED. FOR MODIFIED (MOD) RUNS, THE RNUMBER`);
WRITELN(PR,`THAT INTERRUPTS A RUN IS THROWN OUT. IN MOD RUNS, THE PROBAB`);
WRITELN(PR,`OF A RUN OF LENGTH R IS  $[1/R! - 1/(R+1)!]$  . A CHI SQUARE TEST`);
WRITELN(PR,` IS PERFORMED USING THE DIFS BETWEEN ACTUAL AND EXPECTED MOD`);
WRITELN(PR,`RUNS. ONLY RUNS FROM 1 TO 5 AND >5 IN LENGTH ARE EXAMINED`);
WRITELN(PR)

```

```

END;

```

```

(****)
PROCEDURE AUX(UP,ORD:BOOLEAN;VAR CT:INTEGER;VAR LO:REAL;VAR SUM:SUMCASES);
(*UPDATES SUMS FOR ALL CASES OF RUNS AND MODES*)

```

```

VAR L,X :REAL;

```

```

BEGIN
IF UP=TRUE THEN
BEGIN
L:=LO;
X:=XR;
END

```

```

ELSE
  BEGIN
    L:=LO;
    X:=-XR;
  END;
IF X>L THEN CT:=CT+1;
IF (CT=6) OR (IJ=NT) OR (X<L) THEN
  BEGIN
    SUM[CT]:=SUM[CT]+1;
    CT:=1;
  END;
IF (ORD=TRUE) OR (CT>1) THEN LO:=XR
ELSE IF UP=TRUE THEN STUP:=STUP+1 ELSE STDW:=STDW+1
END;

```

```

(****)
PROCEDURE START;
VAR I:INTEGER;

```

```

BEGIN
  FOR I:=1 TO 6 DO
    BEGIN
      SUPOR[1]:=0;
      SUPMD[1]:=0;
      SDWOR[1]:=0;
      SDWMD[1]:=0;
    END;
    CTUPOR:=1;
    CTUPMD:=1;
    CTDWOR:=1;
    CTDWMD:=1;
    XR:=RND;
    LOUPOR:=XR;
    LOUPMD:=XR;
    LODWOR:=XR;
    LODWMD:=XR;
    STUP:=0;
    STDW:=0

```

```

END;

```

```

(****)
PROCEDURE SUMRUNS; (*FORMS THE SUMS OF UP AND DOWN RUNS FOR ORDINARY
AND MODIFIED CASES. IN MOD CASES, THE RNUMBER THAT BREAKS A RUN IS THROWN
OUT. CONSECUTIVE RUNS OF 1 THRU 5 AND >5 ARE ANALYZED*)

```

```

VAR I :INTEGER;

```

```

BEGIN
  START;
  FOR IJ:=2 TO NT DO

    BEGIN
      WRITELN(IJ); (*SCREEN REMINDER*)
      XR:=RND;
      AUX(TRUE,TRUE,CTUPOR,LOUPOR,SUPOR);(*ORDINARY UP RUNS*)
      AUX(FALSE,TRUE,CTDWOR,LODWOR,SDWOR);(*ORDINARY DOWN RUNS*)
      IF STUP>0 THEN
        BEGIN
          CTUPMD:=1;
          LOUPMD:=XR;
          STUP:=0;
        END
      ELSE AUX(TRUE,FALSE,CTUPMD,LOUPMD,SUPMD); (*MODIF UP RUNS*)
      IF STDW>0 THEN
        BEGIN
          CTDWMD:=1;
          LODWMD:=XR;
          STDW:=0;
        END
      ELSE AUX(FALSE,FALSE,CTDWMD,LODWMD,SDWMD); (*MODIF DOWN RUNS*)
    END
  END;

  (****)
  PROCEDURE PRINTANS; (*PRINTS CHISQ COMPUT*)

  BEGIN
    WRITELN(PR);
    PRB:PROB(3,CHIUP/2);
    WRITELN(PR,'DEGREES OF FREEDOM=6');
    WRITELN(PR,'CHISQ UP=',CHIUP:8:4,'      PROB OF CHANCE OCCRC=',PRB:6:4);
    WRITELN(PR);
    PRB:=PROB(3,CHIDW/2);
    WRITELN(PR,'CHISQ DOWN=',CHIDW:8:4,'      PROB OF CHANCE OCCRC=',PRB:6:4)
  END;

  (****)
  PROCEDURE RUNSTEST; (*TEST UP,DOWN RUNS FOR MODIFIED CASES AND TOTAL NUMBER OF
  ORDINARY RUNS*)

  VAR I,J,SUMUP,SUMDW      :INTEGER;
      T,EXPCTUP,EXPCTDW,DIFUP,DIFDW :REAL;

  BEGIN
    HEADING;
    WRITELN(PR,'RESULTS FOR ORDINARY RUNS');

```

```

FOR I:=1 TO 2 DO WRITELN(PR);
WRITELN(PR,'RUN UPSUMS DWSUMS');
FOR I:=1 TO 6 DO WRITELN(PR,I:3,SUPOR[I]:7,SDWOR[I]:7);
FOR I:=1 TO 3 DO WRITELN(PR);
WRITELN(PR,' RESULTS OF MODIFIED RUNS TEST FOLLOW');
WRITELN(PR,'RUN EXPUP  ACTUP  DIF  EXPDW  ACTDW  DIF');
SUMUP:=0;
SUMDW:=0;
CHIUP:=0;
CHIDW:=0;
FOR I:=1 TO 6 DO
  BEGIN
    SUMUP:=SUMUP+SUPMD[I];
    SUMDW:=SUMDW+SDWMD[I];
  END;
FOR I:=1 TO 6 DO
  BEGIN
    T:=1;
    FOR J:=1 TO I+1 DO T:=T*J;
    IF I<6 THEN PRB:=I/T ELSE PRB:=(I+1)/T;
    EXPCTUP:=PRM*SUMP;
    EXPCTDW:=PRB*SUMP;
    DIFUP:=EXPCTUP-SUPMD[I];
    DIFDW:=EXPCTDW-SDWMD[I];
    CHIUP:=CHIUP+DIFUP*DIFUP/EXPCTUP;
    CHIDW:=CHIDW+DIFDW*DIFDW/EXPCTDW;
    WRITE(PR,I:3,EXPCTUP:8:1,SUPMD[I]:6,DIFUP:8:1,EXPCTDW:8:1);
    WRITELN(PR,SDWMD[I]:6,DIFDW:8:1);
  END;
PRINTANS
END;

(***** MAIN PROGRAM *****)

BEGIN
  REWRITE(PR,'PRINTER:'); (*ENABLE PRINTER*)
  WRITE('HOW MANY TRIALS? ');
  READLN(NT);
  SUMRUNS;
  RUNSTEST;
  FOR I:=1 TO 2 DO WRITE(PR,CHR(ORD(12))) (* NEW PAGE*)
END.

```


APPENDIX C

FREQUENCY DISTRIBUTION AND KOLMOGOROFF-SMIRNOV TEST

10,000 VARIATES ARE DISTRIBUTED INTO 100 BINS. EXPECTED FREQUENCY IS 100. "DIF"=EXPECTED-ACTUAL FREQUENCY. 'KOL' IS THE DIFFERENCE BETWEEN EXPECTED AND ACTUAL CUMULATIVE FREQUENCY.

BIN	OBS	DIF	CUM	KOL	BIN	OBS	DIF	CUM	KOL
0	94	6.0	0.0094	0.0006	34	92	8.0	0.3511	-0.0011
1	113	-13.0	0.0207	-0.0007	35	97	3.0	0.3608	-0.0008
2	101	-1.0	0.0308	-0.0008	36	94	6.0	0.3702	-0.0002
3	89	11.0	0.0397	0.0003	37	105	-5.0	0.3807	-0.0007
4	112	-12.0	0.0509	-0.0009	38	108	-8.0	0.3915	-0.0015
5	96	4.0	0.0605	-0.0005	39	110	-10.0	0.4025	-0.0025
6	99	1.0	0.0704	-0.0004	40	94	6.0	0.4119	-0.0019
7	90	10.0	0.0794	0.0006	41	98	2.0	0.4217	-0.0017
8	0	10.0	0.0884	0.0016	42	91	9.0	0.4308	-0.0008
9	86	14.0	0.0970	0.0030	43	88	12.0	0.4396	0.0004
10	122	-22.0	0.1092	0.0008	44	110	-10.0	0.4506	-0.0006
11	99	1.0	0.1191	0.0009	45	110	-10.0	0.4616	-0.0016
12	123	-23.0	0.1314	-0.0014	46	102	-2.0	0.4718	-0.0018
13	99	1.0	0.1413	-0.0013	47	84	16.0	0.4802	-0.0002
14	98	2.0	0.1511	-0.0011	48	110	-10.0	0.4912	-0.0012
15	92	8.0	0.1603	-0.0003	49	106	-6.0	0.5018	-0.0018
16	94	6.0	0.1697	0.0003	50	118	-18.0	0.5136	-0.0036
17	97	3.0	0.1794	0.0006	51	104	-4.0	0.5240	-0.0040
18	109	-9.0	0.1903	-0.0003	52	94	6.0	0.5334	-0.0034
19	101	-1.0	0.2004	-0.0004	53	112	-12.0	0.5446	-0.0046
20	111	-11.0	0.2115	-0.0015	54	107	-7.0	0.5553	-0.0053
21	108	-8.0	0.2223	-0.0023	55	99	1.0	0.5652	-0.0052
22	130	-30.0	0.2353	-0.0053	56	95	5.0	0.5747	-0.0047
23	103	-3.0	0.2456	-0.0056	57	80	20.0	0.5827	-0.0027
24	99	1.0	0.2555	-0.0055	58	115	-15.0	0.5942	-0.0042
25	80	20.0	0.2635	-0.0035	59	97	3.0	0.6039	-0.0039
26	89	11.0	0.2724	-0.0024	60	102	-2.0	0.6141	-0.0041
27	92	8.0	0.2816	-0.0017	61	101	-1.0	0.6242	-0.0042
28	100	0.0	0.2916	-0.0016	62	85	15.0	0.6327	-0.0027
29	91	9.0	0.3007	-0.0007	63	103	-3.0	0.6430	-0.0030
30	102	-2.0	0.3109	-0.0009	64	96	4.0	0.6526	-0.0026
31	97	3.0	0.3206	-0.0006	65	100	0.0	0.6626	-0.0026
32	120	-20.0	0.3326	-0.0026	66	118	-18.0	0.6744	-0.0044
33	93	7.0	0.3419	-0.0019	67	108	-8.0	0.6852	-0.0052

APPENDIX C (CONCLUDED)

FREQUENCY DISTRIBUTION AND KOLMOGOROFF-SMIRNOV TEST

10,000 VARIATES ARE DISTRIBUTED INTO 100 BINS. EXPECTED FREQUENCY IS 100. "DIF"=EXPECTED-ACTUAL FREQUENCY. 'KOL' IS THE DIFFERENCE BETWEEN EXPECTED AND ACTUAL CUMULATIVE FREQUENCY.

BIN	OBS	DIF	CUM	KOL
68	97	3.0	0.6949	-0.0049
69	97	3.0	0.7046	-0.0046
70	98	2.0	0.7144	-0.0444
71	101	-1.0	0.7245	-0.0045
72	86	14.0	0.7331	-0.0031
73	108	-8.0	0.7439	-0.0039
74	101	-1.0	0.7540	-0.0040
75	98	2.0	0.7638	-0.0038
86	90	11.0	0.7727	-0.0027
77	79	21.0	0.7806	-0.0006
78	92	8.0	0.7898	0.0002
79	116	-16.0	0.8014	-0.0014
80	102	-2.0	0.8116	-0.0016
81	136	-36.0	0.8252	-0.0052
82	88	12.0	0.8340	-0.0040
83	106	-6.0	0.8446	-0.0046
84	112	-12.0	0.8558	-0.0058
85	101	-1.0	0.8659	-0.0059
86	102	-2.0	0.8761	-0.0061
87	92	8.0	0.8853	-0.0053
88	91	9.0	0.8944	-0.0044
89	93	7.0	0.9037	-0.0037
90	107	-7.0	0.9144	-0.0044
91	104	-4.0	0.9248	-0.0048
92	81	19.0	0.9329	-0.0029
93	115	-15.0	0.9444	-0.0044
94	79	21.0	0.9523	-0.0023
95	92	8.0	0.9615	-0.0015
96	96	4.0	0.0711	-0.0011
97	100	0.0	0.9811	-0.0011
98	97	3.0	0.9908	-0.0008
99	92	8.0	1.0000	0.0000

APPENDIX D

OBSERVED DISTRIBUTION OF RANDOM PAIRS

	BINS									
	0	1	2	3	4	5	6	7	8	9
0	53	54	34	50	42	50	52	43	38	49
1	58	59	57	47	47	47	44	59	39	50
2	40	52	48	49	51	52	56	47	44	47
3	48	45	50	53	43	49	57	56	59	60
4	45	54	63	49	53	50	54	51	43	50
5	57	50	59	52	60	55	53	41	39	51
6	40	62	54	48	46	46	47	58	60	51
7	57	40	54	45	39	52	48	38	46	44
8	65	62	46	51	54	51	49	61	62	50
9	42	49	52	54	46	52	35	51	42	44

END

FILMED

8-85

DTIC